REMARKS

Applicant is in receipt of the Office Action mailed June 3, 2004. Claims 1 – 64 were rejected. Claims 3, 55, 58, and 62 have been cancelled, and so their rejections are rendered moot. Claims 1, 4, 49, 53, 56, 59, 60, and 63 have been amended to clarify the claimed invention. Thus, claims 1-2, 4-54, 56-57, 59-61, and 63-64 are currently pending in the application. Further consideration of the present case is earnestly requested in light of the following remarks.

Applicant respectfully notes the provisional obviousness-type double patenting rejection of claims 1 – 64. Once the patent application or one of the co-pending patent applications issues, Applicant will consider filing a terminal disclaimer in the application that is still pending.

The provisional application number listed on page 1 was incorrect. Applicant has changed the first paragraph to specify the correct provisional application number (60/149,942). Applicant respectfully notes that a new oath or declaration is required because the listed provisional application number was incorrect. Applicant is in the process of procuring a new oath or declaration and will submit it in a supplemental amendment.

Section 102 Rejections

Claims 1, 49, 53, 56, and 60 were rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 5,920,718 to Uczekaj et al. (hereinafter referred to as "Uczekaj"). Applicant respectfully traverses this rejection.

As the Examiner is certainly aware, anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim. *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The identical invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).

Moreover, an 'anticipating' reference must describe all of the elements and limitations of the claim in a single reference, and enable one of skill in the field of the invention to make and use the claimed invention. *Bristol-Myers Squibb Co. v. Ben Venue Labs., Inc.*, 246 F.3d 1368, 1378-79 (Fed. Cir. 2001); *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226 (Fed. Cir. 1989)." *In re Merck & Co., Inc. v. Teva Pharm. USA, Inc.*, 347 F.3d 1367, 1372 (Fed. Cir. 2003).

The Office Action indicated that Applicant's argument that the reference fails to teach "a graphical program, where a graphical program includes a plurality of interconnected nodes that visually indicate the functionality of the program" did not overcome the rejection because the features relied upon were not recited in the rejected claims. Applicant has amended the independent claims to explicitly include this feature, and so respectfully submits that Uczekaj does not teach or suggest the invention as represented by the amended independent claims and those claims respectively dependent thereon.

Furthermore, as noted in the previous Response, Uczekaj relates generally to a system for automatically generating application program shell code in an object-oriented system. The system allows a user to enter object information and associated control information. The system automatically generates application program shell code according to the operating system environment in which the system is running and object and object control information entered by the user (Abstract).

Applicant submits that Uczekaj does not teach or suggest numerous features of claims 1, 49, 53, 56, and 60. For example, claim 1 relates to a method for programmatically generating a new graphical program. However, Uczekaj does not even teach or suggest the concept of a graphical program. The Office Action cites Col. 10, lines 7 – 22, asserting that Uczekaj discloses the concept of programmatically generating a new graphical program. However, as also noted previously, this portion of Uczekaj actually pertains to a state diagram for a class defined by the user.

The Office Action asserts that Uczekaj's displaying of a state diagram by activating a display state diagram button is somehow equivalent to programmatically

generating a graphical program as defined in claim 1. Applicant respectfully submits that a state diagram for a class is not at all the same as a graphical program. Furthermore, in Uczekaj's system, the user provides object information and associated control information (Abstract), from which the graphical control system (where "graphical" refers to the GUI, *not* to a graphical program) generates the (text-based) application program shell code, and the state diagram. Applicant notes that the state diagram is simply a display of the object and control information provided by the user, e.g., object names, state names, and state transition method names, and that no actual functionality is included in the state diagram display. Thus, Applicant submits that Uczekaj does not teach all the features and limitations of claim 1, and in fact, teaches away from claim 1.

For at least the reasons provided above, Applicant submits that claim 1 is patentably distinct over Uczekaj, and further submits that claim 1 is non-obvious over Uczekaj, and is thus allowable. Claims 2, and 4-52 depend from claim 1 and so are also allowable. Claims 53, 56, and 60 recite similar features as claim 1, and so Applicant submits that these claims, and those claims respectively dependent thereon, are also patentably distinct and non-obvious over Uczekaj, and are thus allowable.

Claims 1, 49, 53, 56, and 60 were also rejected under 35 USC 102(e) as being anticipated by Morris et al. (US Patent 5,862,372, "Morris"). Applicant respectfully disagrees.

The Office Action asserts that Morris teaches programmatically generating a graphical program, citing the Abstract of Morris which recites "Development of a complete application is accomplished by visually arranging, ordering and interconnecting the objects without the necessity of writing any code". Applicant submits that the 'code' referred to by Morris is traditional text-based code, not graphical program source code (graphical program objects or nodes), and that, like Kodosky, Morris describes *manual creation* of a graphical program, *not* programmatic generation, as clearly described and defined in the present application. This distinction is explicitly made by Morris in the

passage recited below (col. 5, lines 13-22), and Morris goes on to state, "As the author proceeds in developing the application, the system adds to the script". Thus, the author (developer) develops the application (the block diagram or graphical program) manually, then the system automatically generates a script corresponding to the block diagram.

For example, the Morris patent at column 3 lines 29-35 refers to a user directly selecting and dragging icons onto the screen in a certain order to specify a sequence of operations or "script", and Morris states at column 3 line 32 that "Icons representing the objects (and accordingly their functionalities) may be placed (dragged) into an appropriate view. The Morris et al. patent further describes this operation at col. 5 lines 13 – 22:

> During the development of an application with the system of this invention, the user directs the incorporation of objects into one of the four views, modifies the properties associated with that object, and interconnects the objects so chosen. . . . As the system responds to the user's directions, it creates a script representing the application being developed by the user. When the development process is complete, the user stores the script on one of the system's storage devices.
> One of the interesting features of the object oriented authoring system of this invention is that no traditional looking code, written in a programming language, is generated by the system. Rather, the output of the authoring system is a listing of the objects in order of appearance (in the program sequence) along with the properties associated with that object at that place in the sequence. The properties are the information or settings which specify to the object how it will perform. The computer implemented application development system records the settings for the objects and the order of execution of the objects in an output script. Thus, the system operates only at the object level. As the author proceeds in developing the application, the system adds to the script.

Thus the Morris patent describes operation where a user manually selects objects or icons and interconnects them together to specify a script. Note that Morris's system does *not* programmatically generate a *graphical program*, but rather the user manually creates the graphical program (interconnected objects), and the system generates a corresponding script *based on the graphical program*, where "as the author proceeds in *developing the application* (the graphical program), *the system adds to the script.*

Thus, the cited portion of Morris is comparable to the prior art method of a user manually creating graphical programs as described in the Background section of the present application, which recites

> ...a user typically creates a graphical program within a graphical programming environment by interactively or manually placing icons or nodes representing the desired blocks of functionality on a diagram, and connecting the icons/nodes together to represent one or more of the data flow, control flow, and/or execution flow of the program.

In other words, the teaching of Morris is analogous to this prior art method of manually creating a graphical program, except that a script is then created based on the sequence of objects that the user selects and interconnects. In other words, in the Morris patent, the user manually assembles a block diagram (as shown in Figure 2 of Morris) in order to specify a script, whereas in the system of the present application, the user first provides input specifying functionality of the graphical program, and then a software program (the GPG program) executes to programmatically generate a graphical program. Applicant notes that the programmatically generated script in Morris is *not* a graphical program. Significantly, Morris does not teach or suggest any type of programmatic generation of a graphical program. Rather, Morris teaches that a user is required to manually assemble a block diagram.

As for the Examiner's citation of Morris that reads "Development of a complete application is accomplished by visually arranging, ordering and interconnecting the objects without the necessity of writing any code", Applicant submits that, as noted above, the "code" referred to is traditional text-based program code, *not* a block diagram.

Thus, Applicant submits that Morris simply does not teach or suggest "programmatically generating a graphical program". Morris refers to the user actually dragging the objects into the appropriate view, which is not a programmatic creation of a graphical program, but rather is a manual creation. This distinction is emphasized by the further limitations of amended claim 1 that "wherein the information does not specify specific objects for the new graphical program".

For at least the reasons provided above, Applicant submits that claim 1 is patentably distinct over Morris, and further submits that claim 1 is non-obvious over Morris, and is thus allowable. Claims 53, 56, and 60 recite similar features as claim 1,

and so Applicant submits that these claims are also patentably distinct and non-obvious over Morris, and are thus allowable.

Removal of the 102 rejection of claims 1-2, 4-54, 56-57, 59-61, and 63-64 is requested.

Section 103 Rejections

Claims 1 –64 were rejected under 35 U.S.C. 103(a) as being anticipated by U.S. Patent No. 5,732,277 to Kodosky et al. (hereinafter referred to as "Kodosky") in view of U.S. Patent No. 5,920,718 to Uczekaj et al. (hereinafter referred to as "Uczekaj"). Applicant respectfully traverses this rejection.

As the Examiner is certainly aware, to establish a prima facie obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. In re Royka, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. Obviousness cannot be established by combining or modifying the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion or incentive to do so. In re Bond, 910 F. 2d 81, 834, 15 USPQ2d 1566, 1568 (Fed. Cir. 1990).

Moreover, as held by the U.S. Court of Appeals for the Federal Circuit in Ecolochem Inc. v. Southern California Edison Co., an obviousness claim that lacks evidence of a suggestion or motivation for one of skill in the art to combine prior art references to produce the claimed invention is defective as hindsight analysis.

In addition, the showing of a suggestion, teaching, or motivation to combine prior teachings "must be clear and particular . . .. Broad conclusory statements regarding the teaching of multiple references, standing alone, are not 'evidence'." *In re Dembiczak*, 175 F.3d 994, 50 USPQ2d 1614 (Fed. Cir. 1999). The art must fairly teach or suggest to one to make the specific combination as claimed. That one achieves an improved result by making such a combination is no more than hindsight without an initial suggestion to make the combination.

Applicant respectfully submits that neither Kodosky nor Uczekaj provides a motivation to combine, and so the combination is improper. For example, nowhere does

Kodosky teach, suggest, or even mention, programmatically generating a graphical program. Similarly, Uczekaj nowhere mentions or suggests a graphical program at all, but rather, describes generation of application program shell code, i.e., a *program shell*, which Uczekaj itself distinguishes from an application program. Thus, Uczekaj does *not* disclose programmatic generation of an application (program), and specifically does not disclose or even hint at, programmatic generation of a graphical program. In fact, the only diagram Uczekaj describes is a state diagram for a class, which is specifically *not* a graphical program, as defined in the present Specification and Claims. Applicant further submits that even were Kodosky and Uczekaj properly combinable, which Applicant argues they are not, the resulting combination would not produce Applicant's invention as claimed.

For example, the Office Action asserts that Kodosky teaches programmatically generating a new graphical program, including executing a graphical program generation (GPG) program to programmatically generate the new graphical program based on received information, citing col. 16, lines 61-67, col. 13, lines 40-62, col. 46, lines 48-61, col. 3, lines 56-60, col. 8, lines 3-12, col. 17, lines 36-41, col. 2, lines 63-67, and col. 4, lines 28-39. Applicant respectfully disagrees.

Applicant respectfully submits that the Examiner has interpreted the term "graphical program" and "executable program" incorrectly, and more specifically appears to consider them to be the same thing, when in actuality they are quite distinct.

For example, amended claim 1 recites in part:

> the GPG program receiving information, wherein the information specifies functionality of the new graphical program, wherein the information does not specify specific objects for the new graphical program;
>
> the GPG program programmatically generating the new graphical program in response to said information specifying the functionality of the new graphical program, wherein the new graphical program implements the specified functionality, and wherein the new graphical program comprises a plurality of interconnected nodes that visually indicate the functionality of the new graphical program.

Thus, the present application relates to a first program—a GPG program, programmatically (automatically) generating a *graphical program* based on received information, where the generated graphical program comprises *a plurality of interconnected nodes that visually indicate the functionality of the new graphical program*, referred to as a block diagram. In other words, information specifying functionality of the graphical program is received, and the GPG program *programmatically* generates the graphical program, e.g., the block diagram.

In contrast, Kodosky teaches *a data flow diagram being assembled in response to user input specifying objects to add to the data flow diagram.* In other words, the user interactively, i.e., *manually*, creates a data flow diagram to represent the desired execution procedure. Execution instructions for executing the procedure represented by the data flow diagram are constructed (Abstract; Col 7, line 66 – Col 8, line 12). It should be noted that the execution instructions of Kodosky are based on, or underlie, the graphical program, much as machine executable code underlies a C program, where the machine executable code is generated by a compiler or interpreter. Thus, in the system of Kodosky, *the user creates the graphical program.* For example, the user may drag and drop graphical program nodes onto a block diagram and interconnect or "wire" the nodes together, thereby manually generating the block diagram. This is in direct contrast to the present invention, where the block diagram is generated programmatically. This distinction is further emphasized by the inclusion (in the previous Response) of the limitation "wherein the information specifies functionality of the new graphical program, wherein the information does not specify specific objects for the new graphical program". In other words, the amended claim specifies that the user does *not* specify or otherwise select the graphical program objects (e.g., graphical program elements, icons, or nodes) that make up the graphical program or block diagram.

Thus, Kodosky does not teach or suggest the concept of programmatically generating a graphical program at all, and even more specifically does not disclose programmatic generation of the graphical program without receiving input that specifies specific objects for the graphical program.

The Examiner cites Uczekaj in an effort to correct the deficiencies of Kodosky, asserting that Uczekaj discloses a system "to allow a user to enter various specific objects defined by the user", by providing "information not specifying specific objects", allowing "a user to enter objects with state information and, based on the entered information", generating "application shell code". Applicant respectfully disagrees.

Applicant submits that Uczekaj fails to correct the deficiencies of Kodosky. For example, as noted above, Uczekaj fails to teach a graphical program at all "wherein the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program", and more specifically fails to teach or suggest programmatically generating a graphical program. Thus, the Uczekaj reference does not teach or suggest a graphical program, but rather, teaches generation of program shell code (in a text-based programming language) for different operating systems. As Uczekaj states:

> "The generated code is called application shell code because all code is generated except the specific code for any user method names entered in define user method section 274 and any transition method names entered in define transition method section 284. The specific code for these two types of methods must be entered into specific locations within the application shell code. The user may either enter user and transition method code in the graphical interface tool through an edit function or outside the graphical interface tool through a file editor that allows a programmer to edit and enter code." (col. 10, lines 27-37)

Thus, in the system of Uczekaj, the actual program functionality must be manually written by the user, i.e., only the *shell* of the program is generated programmatically, and the program shell is itself written in a text-based programming language. Uczekaj is quite clear in making the distinction between an application (program) and an application shell (program shell), as may be seen in the paragraph cited above. Thus, not only is Uczekaj's program shell code not a graphical program, but it is also not a functional program, but rather requires the user to provide the functionality. Thus, as noted above, no graphical program is generated at all in Uczekaj, but only a text-based program shell, to which the user must add his own (text-based) program code, i.e., the functionality must be provided by the user.

Applicant respectfully reminds the Examiner that it is insufficient to select from the prior art the separate components of the inventor's combination, using the blueprint supplied by the inventor (*Interconnect Planning Corp. v. Feil*, 774 F.2d 1132, 1143, 227 USPQ 543, 551 (Fed. Cir. 1985), and that the prior art must suggest to one of ordinary skill in the art the desirability of the claimed combination (*Fromson v. Advance Offset Plate, Inc.*, 755 F.2d 1549, 1556, 225 USPQ 26, 31 (Fed. Cir. 1985). Applicant submits that the Examiner has simply extract portions of each cited reference that the Examiner believes correspond to particular aspects of Applicant's invention as claimed, and that not only is this improper (and is hindsight analysis), but that even in combination, the cited portions do not produce Applicant's invention as represented in claim 1, as shown above. Moreover, Applicant submits that Uczekaj (as well as Kodosky) actually *teaches away* from Applicant's invention. For example, claim 1 does not include Uczekaj's feature of generating (text-based) application program shell code, nor generating a state diagram for a class, nor does claim 1 include Kodosky's manual creation of a graphical program.

Thus, neither Kodosky nor Uczekaj, either singly or in combination, teaches or suggests the limitations of claim 1, and so, for at least the reasons provided above, Applicant submits that amended claim 1 is patentably distinct and non-obvious over the cited art, and is thus allowable. Claims 2 and 4-48 are dependent upon claim 1 and are thus also believed to be allowable for at least this reason.

Independent claims 49, 56, and 60 include similar features and limitations as claim 1, and so the above arguments apply with equal force to those claims. Thus, Applicant submits that claims 49, 56, and 60 are patentably distinct and non-obvious over the cited art, and so claims 49, 56, and 60, and those claims respectively dependent thereon, are also allowable.

Additionally, various of the independent claims have been amended to overcome rejections under 35 U.S.C. 102, and, as argued above, the amended independent claims are nonobvious and are allowable, as well. Applicant respectfully submits: "If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988)" as

stated in the MPEP §2143.03. With this in mind, Applicant respectfully submits that claims 2, 4-52, 54-55, 57-59, and 61-64 are allowable.

Removal of the 103 rejection of claims 1-2, 4-54, 56-57, 59-61, and 63-64 is requested.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 50-1505/5150-44100/JCH.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

☐ Request for Approval of Drawing Changes

☐ Notice of Change of Address

☐ Check in the amount of $ for fees ( ).

☐ Other:

Respectfully submitted,

Mark S. Williams
Reg. No. 50,658
AGENT FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: Sept. 1, 2004 JCH/MSW